CP/M*

Versions 1.4 & 2.X

Programmer's Reference Guide

REVISED EDITION BY SOL LIBES,
Editor of *Microsystems*

*CP/M is a registered trademark of Digital Research.

BUILT-IN COMMANDS

```
DIR                    Display file directory {current drive
DIR d:                                        {designated drive
DIR filename.typ     Search for named file, current drive
DIR *.typ            Display all files of named type,curr drv
DIR filename.*       Display all types of designated filename
DIR x????.*        Display all filenames 5 characters
                          long and start with letter x

TYPE filename.typ                  Display ASCII file {current drive
TYPE d:filename.type                                  {designated drive

ERA filename.typ                  {named file, current drive
ERA *.*                           {all files, curr drv, V2.x curr user
ERA *.typ                  Erase  {all files } designated {type
ERA d:filename.typ                {named file}            {drive
ERA filename.*                    {all types of named file, curr drv

REN nuname.typ=olname.typ  } RENname file {current drive
REN d:nuname.typ=olname.typ}              {designated drive

SAVE n filename.typ                SAVE as named file {current drive
SAVE n d:filename.typ                                 {designated drive
                          n pages (page=256 bytes) start @ 100H

d:                        Switch to designated disk drive
                            A-D V1.4; A-P V2.x
USER n                    Change user area (Version 2.x)
```

ED COMMANDS

```
nA      Append n lines to buffer (n=0 -use half of buffer)
B                              {beginning}
-B      Move pointer to        {end       } of file
nC                             {forward n characters
nD      Delete n characters forward
E       End edit, close file, return to CP/M
nFs     Find n-th occurrence of string 's'
H       end edit, move pointer to beginning of file
I       Insert text at pointer until ^Z typed
Is      Insert string at pointer
nK      Kill n lines starting at pointer
nL      move pointer n lines
nMx     execute command string 'x' n times
nNs     global F-command- until end of file
O       abort ED, start over with original file
nP      list next n pages of 23 lines (n=0 -current page)
Q       Quit without changing input file
Rfn     Read fn.LIB into buffer at current pointer
nSx^Zy  Substitute string 'y' for next n forward
           occurrences of string 'x'
nT      Type n lines
U       change lower case to upper case (next entry)
V       enable internal line number generation
nW      Write n lines to output file (start at
           beginning of buffer)
nX      Write next n lines to file 'X$$$$$$$.LIB'
nZ      Pause n/2 seconds (2MHz)
n                          {n lines}
<CR>    Move {forward      {1 line } and type one line
 -          {backward                }
n:x     move to n line number and perform 'x' command
:mx     perform command 'x' from current line to line m
n::mx   move to n line number and perform command 'x'
        through line number m

note: "-" valid on all positioning and display commands
           for backward movement (e.g. -nC)
```

PIP COMMANDS

```
PIP                        Initiate Peripheral Interchange Program
*d:=s:filename.typ           Copy named file       {from source drv
*d:nuname.*=s:olname.typ   Copy&change filename    {to destinat drv
PIP d:=s:filename.typ      Initiate PIP and copy  named file
PIP d:=s:*.*               from source drv  {all files
PIP d:=s:filename.*                   to    {all named files
PIP d:=s:*.typ             destination drv  {all files named typ
PIP LST:=filename.typ                                {list device
PIP PUN:=filename.typ      send named file to        {punch device
PIP CON:=filename.typ                                {console device
PIP filename.typ=RDR:      Copy data from reader device to
                                named file (current drive)
*nuname.typ=aname.typ,bname.type,cnametyp{ ASCII     } copy&con-
*d:nuname.type=s:aname.typ,s:bname.typ    {          } catenate
*nuname.typ=aname.typ[X],bname.typ[X]     {non-ASCII } files
PIP LST:=aname.typ,bname.typ          send files in sequence
PIP LST:=s:name.typ,s:name.typ         to list device
```

PIP PARAMETERS

```
[B]   - read data block until ^S character
[Dn]  - delete characters past column n
[E]   - echo all copy operations to console
[F]   - remove form feeds
[Gn]  - get file from n user area - V2.x
[H]   - check for proper hex format
[I]   - same as H plus ignores ":00"
[L]   - change all upper case characters to lower case
[N]   - add line numbers with leading zeros suppressed
[N2]  - same as N plus leading zeros & tab
[O]   - object file transfer;ignores end-of-file
[P]   -{insert form feed every {60}{lines
[Pn]  -{                       {n }
[Qstring^Z] - Quit copying  after }string is found
[Sstring^Z] - Start copying when  }
[R]   - read SYS file (V2.x)
[Tn]  - expand tab space to every n columns
[U]   - change all lower case characters to upper case
[V]   - verify copied data
[W]   - delete R/O files at destination (V2.x)
[X]   - copy non-ASCII files
[Z]   - zero parity bit on all characters in file
```

PIP KEYWORDS

```
CON: CONsole device (defined in BIOS)
EOF: send End-of-File (ASCII-^Z) to device
INP: INPut source (patched in PIP)
LST: LiST device (defined in BIOS)
NUL: send 40 NULls to device
OUT: OUTput destination (patched in PIP)
PRN: same as LST:; tabs every 8th character, numbers
       lines & page ejects every 60 lines with
       initial eject
PUN: PUNch device  } defined in BIOS
RDR: ReaDeR device }

  refer to IOBYTE section for additional physical devices
```

ASM CONVENTIONS

labels followed by colon 1- 6 alphanumeric characters
symbol (eg. EQU) no colon first must be alpha, ? or .

Assembly Program Format (space separates fields)
label: opcode operand(s) ;comment

Operators (unsigned)

a+b	a added to b
a-b	difference between a and b
+b	Ø+b (unary addition)
-b	Ø-b (unary subtraction)
a*b	a multiplied by b
a/b	a divided by b (integer)
a MOD b	remainder after a/b
NOT b	complement all b-bits
a AND b	bit-by-bit {AND, OR, XOR} of a and b
a OR b	
a XOR b	
a SHL b	shift a {left, right} b bits, end off, zero fill
a SHR b	

Hierarchy Of Operations
highest: * / MOD SHL SHR
- +
NOT
AND
lowest: OR XOR

Constants
Numeric (post radix)
B=binary
Ø,Q=octal
D=decimal(default)
H=Hexidecimal
ASCII - in quotes (e.g. 'A')

Pseudo-ops

ORG const	Set program or data origin (default=Ø)
END start	End program. Optional address where execution begins
EQU const	Define symbol value(may not be changed)
SET const	Define symbol value(may be changed later)
IF const	Assemble block conditionally until ENDIF
ENDIF	Terminate conditional assembly block
DS const	Define storage space for later use
DB byte[,byte...,byte]	Define bytes as numeric or ASCII constants
DW word[,word...,word]	Define word(s) (two bytes)

const=constant (true if bit-Ø=1 otherwise false)

ASM ERROR CODES

D	Data error (element cannot be placed in data area)
E	Expression error (ill-formed expression)
L	Label error
N	Not implemented
O	Overflow (expression too complicated to compute)
P	Phase error (label has different values on each pass)
R	Register error (specified value not compatible with op code)
U	Undefined label (label does not exist)
V	Value error (operand improper)

TRANSIENT COMMANDS

```
DDT                          Initiate Dynamic Debugger Tool program
DDT filename.typ             Initiate DDT and load named file

ASM filename                 Assemble named ASM {current drive
ASM d:filename                  file on:        {designated drive
ASM filename.abc              a=source file drv; b=HEX file destin-
                              ation drv (Z=skip);c=PRN file destin-
                              ation drv (X=console,Z=skip)
LOAD filename                Make .COM file from {current drive
LOAD D:filename                named HEX file on:{designated drive

DUMP filename.typ            Display file in hex {current drive
DUMP d:filename.typ                              {designated drive

MOVCPM n                                 {and execute nKbyte CP/M system
MOVCPM n *                   Create      {image of nKbyte CP/M system
MOVCPM * *                               {image of maxKbyte CP/M for
                                                 SYSGEN or SAVE
SYSGEN                       Initiate SYStem GENerate program

SUBMIT filename parameters   Execute SUB file using optional
                                       parameter(s)
XSUB                         Execute eXtended SUBmit program (V2.x)

ED filename.typ              Execute EDitor program to create
ED d:filename.typ                         or edit named file

STAT                         Display STATus-R/W or R/O {current drv
STAT d:                       and available disk space {named drive
STAT DEV:                                 {DEVice assignments
STAT VAL:                                 {VALid device assignments
STAT DSK:                    Display      {DiSK characteristics}
STAT USR:                                 {current USeR areas  } V2.x
STAT filename.typ $S                      {size of file        }
STAT filename.typ                         {file characteristics{curr drv
STAT d:filename.typ                                            {named drv
STAT d:=R/O                              {designated drive to Read-only
STAT filename.typ $R/O                   {                    {Read-only
STAT filename.typ $R/W       Change      {                    {Read-Write} V2.x
STAT filename.COM $SYS                   {named file to       {System file}
STAT filename.COM $DIR                                        {Drctry file}
STAT gd:=pd:                   Change general device (CON:,LST:,PUN:
                                  and/or RDR:) assignment of
                                  physical device (see IOBYTE)
```

CP/M DISK FORMAT

Media: 8" soft-sectored floppy-disk single density
(IBM 3740 standard)
Tracks: 77 (numbered 0 thru 76)
Sectors/Track: 26 (numbered 1 thru 26)
Bytes/Sector: 128 data bytes (one logical record)
Storage/Disk: 256,256 bytes (77*26*128)
File Size: any number of sectors from zero to capacity of disk.
Extent: 1Kbytes-8 sectors (smallest file space allocated)
Skew: 6 sectors standard (space between consecutive physical sectors on track): 1-7-13-19-25-5-11-17-23-3-9-15-21-2-8-14-20-26-6-12-18-24-4-10-16-22

System: Track 0 & 1 (optional)
Track-0,sector 1: boot loader
Track-0,sectors 2-26: } CCP & BDOS
Track-1,sectors 1-17: }
Track-1,sectors 18-26: CBIOS

Directory: Track 2: 16 sectors typ. 32-bytes/entry
(64 entries typ.) - extents-0 and 1

User File Area: Remaining sectors on Track-2 and -3 to 76
Extents 2 and above

COMMAND CONTROL CHARACTERS

charac	function	ASCII code
C	Reboot CP/M (warm boot)	03H
E	Start new line	05H
H	Backspace and delete (V2.x)	08H
I	Tab 8 columns	09H
J	Line feed	0AH
M	Carriage return	0DH
P	Printer on/printer off	10H
R	Retype current line	12H
S	Stop display output - any character execpt ^c restarts output	13H
U	Delete line	15H
X	same as ^U (V1.4) backspace to start of line (V2.x)	18H
Z	End of console input (ED & PIP)	1AH
delete rubout	Delete and display last character (tape only)	7FH 7FH

IOBYTE (0003H)

Bit Position / Device		LST: 7 6	PUN: 5 4	RDR: 3 2	CON: 2 1
Dec	Binary				
0	00	TTY:	TTY:	TTY:	TTY:
1	01	CRT:	PTP:	PTR:	CRT:
2	10	LPT:	UP1:	UR1:	BAT:
3	11	UL1:	UP2:	UR2:	UC1:

TTY: TeleTYpe
CRT: Cathode Ray Tube type terminal
BAT: BATch process(RDR=input,LST=output)
UC1: User defined Console
LPT: Line Printer
UL1: User defined List device
PTR: Paper Tape Reader
UR1:, UR2: User defined Reader devices
PTP: Paper Tape Punch
UP1:, UP2: User defined Punch devices

FILE TYPES

ASC ASCII text file, usually Basic source
ASM ASseMbly language file (source for ASM program)
BAK BAcKup copy file (created by editor)
BAS BASic source program file, usually tokenized
COM COMmand file (transient executable program)
DAT DATa file
DOC DOCument file
FOR FORtran source program file
INT INTermediate Basic program file (executable)
HEX HEXadecimal format file (for LOAD program)
LIB Library file used by macro assembler
PLI PL/I source file
PRN PRiNt file (source and object produced by ASM)
REL RELocatable module
SAV System file (V2.x)
SUB SUBmit text file executed by SUBMIT program
SYM SID symbol file
TEX TEXt formatter source file
XRF Cross reference file
$$$ Temporary file

Filename - 8 characters maximum
Filetype - 3 characters maximum

Invalid filename and filetype characters:
< > . , ; : = ? [ ]

DDT COMMANDS

```
A sad              Assemble symbolic code ; start at sad

D                  Dump RAM    {cad; 16 lines
D sad              to console  {sad; 16 lines
D sad,ead          from:       {sad thru ead

F sad,ead,const Fill RAM from sad thru ead with constant

G                  Start       {saved PC
G sad              program     {sad
G sad,bp1          execution   {sad and stop at bp1
G sad,bp1,bp2      at:         {sad and stop at bp1 or bp2
G,bp1,bp2                      {cad and stop at bp1 or bp2

H a,b              Display hex a+b and a-b

I filename         Set up FCB   {user code
I filename.typ      (5CH) for:  {R-command (HEX or COM file)

L                  Dissasemble {cad; 12 lines
L sad              RAM         {sad; 12 lines
L sad,ead          from:       {sad thru ead

M sad,ead,nad   Move RAM block from sad thru ead to nad

R                  Read file specified by I command to RAM at
R offset             normal address + optional offset

S sad              Substitute into RAM starting at sad

T n                Execute n instructions (default=1) with
                         register dump (trace)
U n                Execute n instructions (default=1) with
                         register dump after last instruction
Xr                 Examine/change registers or flags
X                  Examine registers (flag reg:C=carry, Z=zero,
                       M=sign, E=parity, I=aux carry)

     cad=current address       sad=start address
     nad=new address           ead=end address
     ?=error, can mean: file cannot be opened,checksum error
         in HEX file or Assembler/Dissasembler overlayed.
```

LOGIN BYTE (0004H)

```
low  nibble = current drive (0=A,1=B,etc.)
high nibble = current user  (V2.x only)
```

FILE CONTROL BLOCK

Byte(s)		function		
0	dr	Drive code (0=current, 1=A, 2=B, etc)		
1-8	f1-f8	File Name		
9-11	t1-3	File Type t1=1-R/O; t2=1-SYS		
12	ex	current EXtent number		
13	s1	reserved	V1.4	not used
14	s2	=0 on BDOS call to Open,Make,search	V1.4	always 00H
15	rc	extent Record Count		
16-31	d0-dn	Disk map		
32	cr	current record for r/w		
33-35	rn	random record number		

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
dr	f1	f2	f3	f4	f5	f6	f7	f8	t1	t2	t3	ex	s1	s2	rc

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
d0	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	d11	d12	d13	d14	d15	cr	r0	r1	r2

MEMORY ALLOCATIONS

(b=memsize-20K V2.x; memsize-16K V1.4)

	Hex Memory Locations	Contents
System Scratch Area (0-FFH)	0-2	jump to BIOS warm start entry point
	3	IOBYTE
	4	login drive number and current user
	5-7	jump to BDOS
	8-37	reserved: interrupt vectors & future use
	38-3A	RST7-used by DDT or SID programs
	3B-3F	reserved for interrupt vector
	40-4F	scratch area used by CBIOS
	50-5B	not used
	5C-7C	File Control Block (FCB) area (default)
	7D-7F	Random record position-V2.x (default)
	80-FF	DMA buffer area (128 bytes) for input and output (default)

Transient Program Area	100...33FF+b	COM file area	V2.x
	100...28FF+b		V1.4
CCP area	3400+b-3BFF+b	Console Command Processor	V2.x
	2900+b-30FF+b		V1.4
BDOS area	3C00+b-49FF+b	Disk Operating System	V2.x
	3100+b-3DFF+b		V1.4
BIOS area	4A00+b-4FFF+b	I/O system	V2.x
	3E00+b-3FFF+b		V1.4

BIOS ENTRY POINTS

Hex addr	Vector Name	Function	Value Passed	Value Returned
**00	BOOT	cold start entry point		C=0
**03	WBOOT	warm start entry point		C=drv no
**06	CONST	check for console ready		A=const
**09	CONIN	read from console		A=chara
**0C	CONOUT	write to console	C=chara	
**0F	LIST	write to list device	C=chara	
**12	PUNCH	write to punch device	C=chara	
**15	READER	read from reader device		A=chara
**18	HOME	move head to track-0		
**1B	SELDSK	select drive	C=drv no	HL=dph*
**1E	SETTRK	set track number	C=trk no	
**21	SETSEC	set sector number	C=sec no	
**24	SETDMA	set DMA address	BC=DMA	
**27	READ	read selected sector		A=dskst
**2A	WRITE	write selected sector		
**2D*	LISTST	get list status		A=lstst
**30*	SECTRAN	sector translate subroutine	BC=lsecno DE=smap	HL=pysec

const=console status
00=idle
FF=data avail
dph=disk parameter/
header address
dskst=disk status
00=OK
01=error
lstst=list status
00=busy
FF=ready

lsecno=logical sector number
pysec=physical sector number
smap=sector interlace map
address
chara=character
drv no=drive number
trk no=track number
sec no=sector number
DMA=DMA address
* not used in V1.4
**= contents of location 0002H

BDOS FUNCTION CALLS

(request to BDOS to perform specified functions)

	Function Number in C reg Dec	Hex	Function	Value Passed to BDOS in DE(or E)regs	Value Returned in A (or HL) regs
Perip-heral I/O	0	00	system reset	--	--
	1	01	console read	--	char
	2	02	console write	E=char	--
	3	03	reader read	--	char
	4	04	punch write	E=char	--
	5	05	list write		--
	6	06	direct con IO (V2.x)	E= FFH(input) char(output)	0=not ready char
	7	07	get IOBYTE	--	IOBYTE
	8	08	set IOBYTE	E=IOBYTE	--
	9	09	print string	string addr	--
	10	0A	read console buffer	addr of data buffer	chars in buffer
	11	0B	get console status	--	00(not ready) FF(ready)
	12	0C	lift head(V1.x)	--	--
			get vers (V2.x)	--	HL=version no.
	13	0D	reset disk **		--
	14	0E	select disk	E=drive no	--
	15	0F	open file		
	16	10	close file	FCB addr	dir
	17	11	search for file		FF(not found)
	18	12	search for next	--	*
Disk I/O	19	13	delete file		
	20	14	read next recrd	FCB addr	00(valid)
	21	15	write next recd		
	22	16	create file		dir FF(disk full)
	23	17	rename file	old file FCB addr	directory code FF(not found)
	24	18	get login vectr	-- (V1.4)	HL=drive code
	25	19	get disk no.	--	A=cdn
	26	1A	set DMA addr.	DMA addr	--
	27	1B	get alloc vectr	--	HL=ava
	28	1C	write protect	--	--
	29	1D	get R/O vector	--	HL=R/O vector
	30	1E	set file attrib	FCB addr	dir
V2.x only	31	1F	get addr (disk parameters)	--	HL=dpba
	32	20	set/get user code	E= FFH(get) user code(set)	current code --
	33	21	read random		error code***
	34	22	write random	FCB addr	
	35	23	compute file size	(r0,r1,r2 format)	random record field set
	36	24	set random rec		
V2.2 & later	37	25	reset drive	drive vector	0
	40	28	write random with zero fill	FCB addr	return code
not used	38	26			
	39	27			

* V1.4 none
** V1.4 intializes system and selects A drive
*** error codes: 01-reading unwritten data
03-cannot close current extent
04-seek to unwritten extent
05-directory overflow (write only)
06-seek past physical end of disk

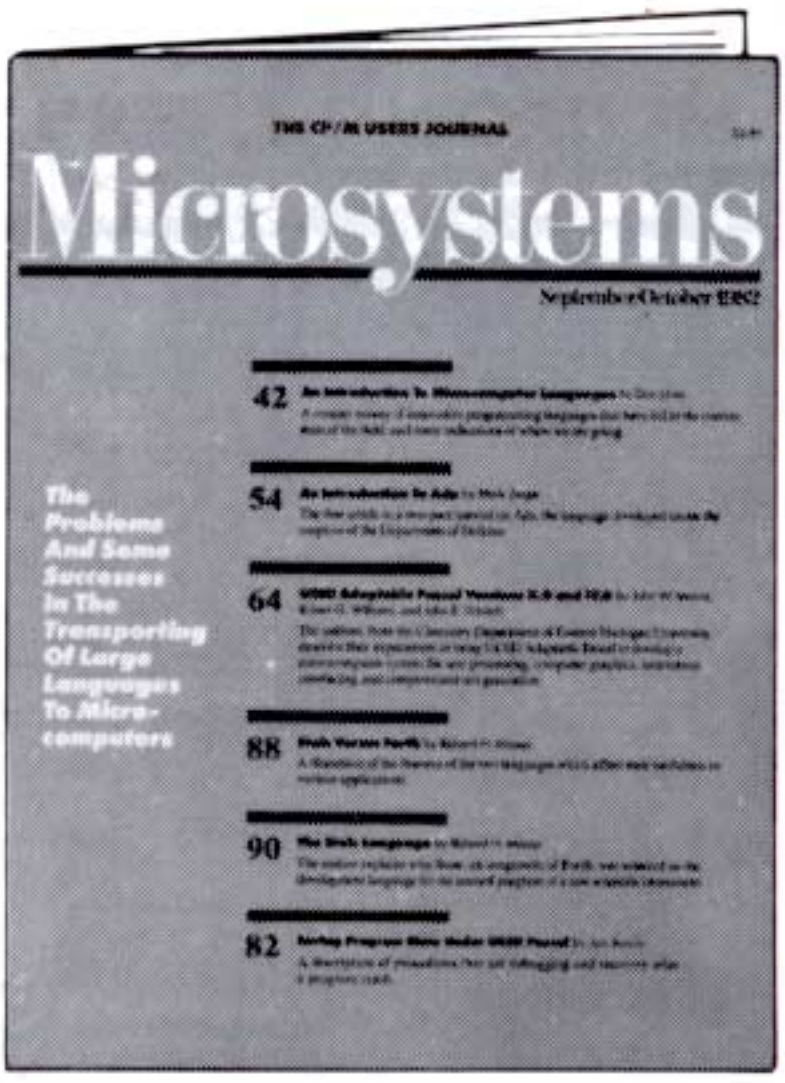
char=character (ASCII)
addr=address
dir =directory code
cdn =current drive number (A=0,B=1,etc)
dpba=disk parameter block address

Microsystems

the advanced systems journal for the serious microcomputer user

MICROSYSTEMS is *not* for beginners or game players. It's the only advanced journal written exclusively for serious, sophisticated programmers and operating systems users—single and multiple. MICROSYSTEMS will keep you up to date with the state of the art in CP/M, CP/M-86 and MP/M systems and compatible programs, MS-DOS, OASIS, UNIX, Xenix and other new generation systems and designs. You'll also find system refinements, operating procedures and innovations, program conversions and certain high-level applications. In addition, MICROSYSTEMS provides an extensive software directory and software and hardware product reviews in each issue.

If you're one of the select few who can take advantage of the information MICROSYSTEMS provides, you owe it to yourself to subscribe. Just punch out the subscription term you prefer on the enclosed card and mail it in the postpaid envelope provided. You'll save up to 33%!



NOTE:

If your profession involves computer usage, your subscription to MICROSYSTEMS may be tax deductible. Check with your accountant.

CP/M and MP/M are registered trademarks of Digital Research Corporation.

OASIS is a trademark of Phase One Systems, Inc.

UNIX is a trademark of Western Electric.

MS-DOS and Xenix are trademarks of Microsoft Inc.

MS-16-1005-GP